On the Use of Don't Cares during Symbolic Reachability Analysis

S. Reda[*], A. Wahba[**], A.Salem[*], D. Borrione[***], M. Ghonaimy[*]

[*]Computer & Systems Dept.
Faculty of Engineering
Ain Shams University
Cairo, Egypt

[**]Mentor Graphics Egypt
Cairo, Egypt

[***]Laboratoire TIMA - INPG
46 Avenue Felix Viallet
38031 Grenoble cedex, France

Abstract
We present a new symbolic algorithm for reachability analysis in sequential circuits. Using don't cares from the computed reachable states, we introduce flexibility in choosing the transition relation, which can be used to minimize its Binary Decision Diagram (BDD). This can reduce the time-consuming image computation step. The technique is implemented and integrated in our equivalence checking system M-CHECK and its efficiency is shown on the ISCAS-89 benchmark circuits.

1. Introduction

Reachability analysis is the process of extracting all the states reachable from the initial state of a given Finite State Machine (FSM). Reachability analysis is the core step in a large number of VLSI CAD problems including sequential circuit verification and synthesis. Symbolic reachability analysis techniques based on BDD's were introduced by Coudert & Madre [1]. The key to the success of their approach is that BDDs are capable of representing a large number of states efficiently. However, in some cases, the BDD of the transition relation of the FSM requires too much memory and thus their approach becomes computationally expensive. Hojati et al. [2] recommended the use of partitioned transition relations and of early quantification scheduling techniques to minimize the time needed to build the transition relation. However, this approach is not efficient if all transition functions share the same support variables. Coudert et al. [1] proposed heuristics to reduce the frontier set BDD using the reached states. Brayton et al. [3] suggested that we should only care about the transitions to the set of states not reached thus far, the don't care transitions could be used to minimize the transition relation BDD.

In this paper, we extend the range of don't cares that can be exploited to minimize the transition relation BDD. More specifically, we propose to add transitions from *all* states to the computed reachable states; these transitions act as an additional source of don't cares. We show that these don't cares combined with the don't cares proposed by Brayton in [3] can further minimize the transition relation BDD size, and accelerate the image computation step. Moreover, with our modification the computation of reachable states always lead to states that never were reached before, which in turn allows us to save the reachable states in disjoint BDDs.

In the following, section 2 gives the basic definitions and notations used throughout the paper. Our proposed approach is explained in section 3, the experimental results are shown in section 4. Section 5 presents our conclusions.

2. Basic Definitions

FSM: A finite state machine M is a 6-tuple $M = (S, \Sigma, \Delta, \delta, \lambda, s^\circ)$. S is the finite set of states, Σ is the input alphabet, Δ is the output alphabet, δ is the state transition function with $\delta: S\times\Sigma\rightarrow S$, λ is the output function with $\lambda: S\times\Sigma\rightarrow\Delta$, S° is the initial state.

Image of a function: If $f:A\rightarrow B$ is a function with domain A and range B. The image of $C \subseteq A$ with respect to f is $img(f, C) = \{y \mid y = f(x) \text{ and } x\in C\}$

Figure 1 shows the Huffman representation of a sequential circuit that sorts the combinational part of the circuit and its storage elements. $\underline{x}$, $\underline{q}$, $\underline{q}'$, $\underline{y}$ denote the input, present state, next state and output vector variables.
A FSM can model the sequential circuit as follows: the input alphabet Σ of the FSM is the set of constant vector values that can be taken by $\underline{x}$, and the same correspondence holds between S and $\underline{q}$, Δ and $\underline{y}$. Likewise δ and λ are vector functions which compute each element of $\underline{q}'$ and $\underline{y}$.

Transition relation: The transition relation of a FSM M is a relation indicating the possible transitions from the present states to the next states. Applied to the model of a sequential circuit, the state transition relation R(q, q') is built [1] as given by equation (1).

$$R(\underline{q}, \underline{q}') = \exists \underline{x} \, (\wedge_{i=1}^{i=|\underline{q}|} (q'_i \Leftrightarrow \delta_i(\underline{x}, \underline{q}))) \qquad (1)$$

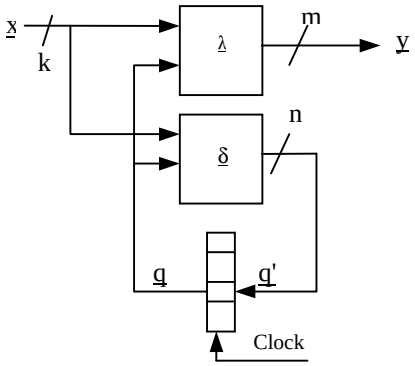

Figure 1: The Huffman model of a sequential circuit

The restrict operator [1] (generalized cofactor [4]) was proposed to minimize BDDs under don't care conditions. The generalized cofactor is based on sibling substitution to reduce the original BDD and it can be best understood by an example. Figure 2a represents the BDD to be minimized, while the BDD in Figure 2b represents the care conditions. Notice that the partial assignment *a = 0, b = 1* is don't care condition and thus, we can replace the node *c* in Figure 2a by its sibling node *d* leading to *c* removal and its parent. The same applies to the assignment *a = 1, c = 1*. The minimized BDD is shown in Figure 2c. However, Hong et al. [5] have shown that sibling substitution can sometimes lead to an increase in the BDD size; thus, they proposed methods to perform safe BDD minimization using don't cares.

3. Proposed Approach

Given the finite state machine model M of a sequential circuit, Figure 3 illustrates the basic breadth first symbolic algorithm for reachability analysis. In the algorithm, Q°, Qr and Qn denote sets of states, and are represented by the BDD of their characteristic function. The performance of the previous algorithm is determined by the efficiency in computing the image of the current states, which is calculated in line number 1. The time complexity of calculating the image is proportional to the product of the sizes of the transition relation BDD (R) and the frontier set BDD (Qn). In this algorithm, the image computation can lead to states that have already been reached before. Figure 4 illustrates a modification of the algorithm in Figure 3, such that the image computation always leads to states that have not been reached before.

```
Forward_Traverse(M)
Let Q° be the initial state of M.
Let Qr be the total reachable states.
Let Qn be the newly reached states.
begin
Qr = Q°; Qn = Qr;
while (true)
 begin
  1. Qn = ∃ q∈ Qn (R(q, q'));
  2. Qn = Qn | q'=q
  3. Qn = Qn ∧ ¬Qr ;
  4. If Qn == ⊥ then break;
  5. Qr = Qr ∨ Qn
 end
end
```

Figure 3: Basic algorithm for reachability analysis

It should be noted that the newly computed relation BDD in line 2 has sometimes a smaller size than the original relation, thus leading to faster execution time of the algorithm. On the other hand, on some benchmarks, line 2 leads to a BDD of larger size. This can be understood since the BDD of the transition relation represents valid (present, next) states. Although the removal of any pair always lead to reduction in the relation size, it is not guaranteed that the BDD produced will be smaller. By this method,

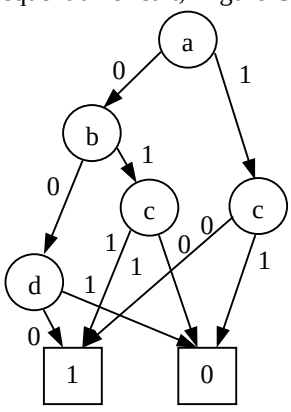

Figure 2a: BDD to be minimized.

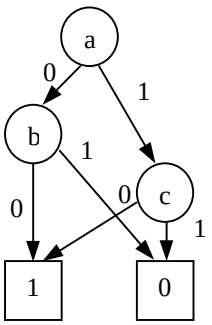

Figure 2b: BDD representing care conditions.

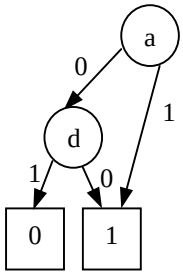

Figure 2c: minimized BDD.

```
Forward_Traverse(M)
begin
Qr = Q°; Qn = Qr;
while (true)
  begin
  1. Qn = ∃ q∈ Qn (R(q, q'));
  2. R(q, q')) = R(q, q')) ∧ ¬ Qn;
  3. Qn = Qn | q' = q
  4.  If Qn == ⊥ then break;
  5.  Qr = Qr ∨ Qn
  end
end
```

Figure 4: Modification of algorithm in Figure 3

we always reach states that were not reached before, therefore, the total reachable states calculated in step 5 need not be always stored in the same BDD (Qr), thus, the reachable states can be stored in multiple disjoint BDDs. Figure 5a illustrates the original relation where *a* is the initial state. Figure 5b represents the relation after the first image computation.

A different modification could be made to the algorithm of Figure 3. The modification is based on the following observation: Adding transitions from all the present states to all the calculated reachable states will not change the total reachable states; since none of the not reachable states is becoming reachable. Thus, we add transitions from *all* states to states already reached before, hoping that this will reduce the corresponding BDD size of the transition relation. The modified algorithm is shown in Figure 6. Figure 7a illustrates the original relation where *a* is the initial state. Figure 7b represents the relation after the first image computation.

From the previous, it can be concluded that we have some flexibility in choosing the new transition relation R_{new}, and this could be summarized by equation 2:

$$(R(q, q') \wedge \neg Qn) \subseteq R_{new} \subseteq (R(q, q') \vee Qn) \quad (2)$$

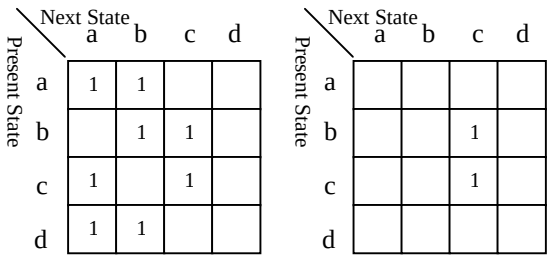

Figure 5a: Original transition relation

Figure 5b: transition relation after first image computation

```
Forward_Traverse(M)
begin
Qr = Q°; Qn = Qr;
while (true)
 begin
  1. Qn = ∃ q∈ Qn (R(q, q'));
  2. R(q, q')) = R(q, q')) ∨ Qn;
  3. Qn = Qn | q' = q
  4. Qn = Qn ∧ ¬Qr ;
  5. If Qn == ⊥ then break;
  6. Qr = Qr ∨ Qn
 end
end
```

Figure 6: Modification of algorithm in Figure 3

R_{new} should be chosen such that its BDD size is the minimum. A method for achieving this is by using the generalized co-factor previously introduced in the basic definitions section. Thus,

$$R_{new} = \text{CoFactor}(R(q, q') \vee Qn, R(q, q') \wedge \neg Qn) \quad (3)$$

The previous function CoFactor() should minimize the BDD of the function $R(q, q')) \vee Qn$ with respect to the care set $R(q, q')) \wedge \neg Qn$. This computation introduces a time consuming step in the forward image computation procedure, but it is hoped that it leads to a decreased transition relation BDD and thus faster execution of line 1. Figure 8 illustrates a conceptual representation of the transition relation. In this representation, all the entries marked with *x* are don't cares and should be utilized to minimize the transition relation BDD using the generalized cofactor. Figure 9 illustrates the final proposed approach.

4. Implementation

The previous ideas have been implemented in C in our equivalence checking system M-CHECK [6].

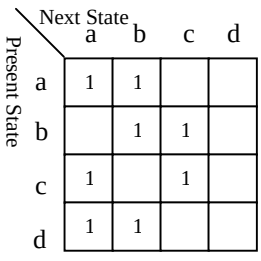

Figure 7a: Original transition relation

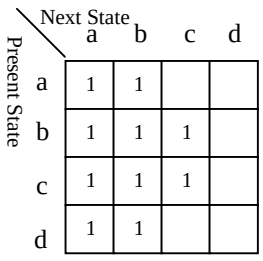

Figure 7b: transition relation after first image computation

Circuit Name	# PI	# PO	# FF	FSM depth	# states	Time (sec)	Time (sec)	Dec.
S27	4	1	2	2	6	0.03	0.01	↓
s298	3	6	14	18	218	0.84	0.72	↓
s444	3	6	21	150	8865	4.9	4.5	↓
s510	19	7	6	46	47	0.05	0.09	
s641	35	24	19	6	1544	4.9	4.94	↓
s713	35	23	19	6	1544	5.9	5.89	
s820	18	19	5	10	25	0.05	0.05	↓
s832	18	19	5	10	25	0.05	0.04	↓
s953	16	23	29	10	505	3.81	1.86	↓
s1196	14	14	18	2	2616	5.83	4.91	↓

Table 1: Reachability analysis using the proposed approach.

Table 1 illustrates the experimental results on some of the ISCAS-89 benchmark circuits [7] using a Sun ULTRA 1. Column 1 is the circuit name. Columns 2, 3, and 4 list the number of primary inputs, primary outputs, and flip-flops respectively. Column 5 lists the depth of the FSM of the circuit. Column 6 lists the number of reachable states. Column 7 gives the time needed for reachability analysis using the algorithm of Figure 1. Column 8 gives the time needed by the proposed heuristics. The symbol ↓ in Column 9 indicates that the proposed heuristics proved successful in decreasing the relation BDD size. Notice that the entries where the BDD was not minimized are due to the unsafe properties of the generalized cofactor [5], [8].

5. Conclusion

In this paper, we proposed heuristics that can be used to extract don't cares in the transition relation during the image computation. We showed how theses don't cares can be utilized to minimize the transition relation BDD and thus the time needed for reachability analysis. The proposed approach was implemented and integrated within our equivalence checker M-CHECK. The efficiency of the proposed algorithm was shown through its application on the ISCAS-89 benchmark circuits.

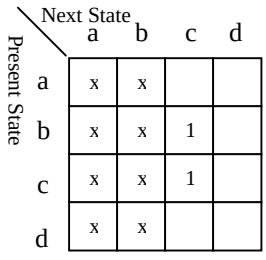

Figure 8: Transition relation with don't cares represented as x.

References

[1] O. Coudert and J.C. Madre, "A unified framework for the formal verification of sequential circuit," In Proc. of Int. conference on Computer-aided Design, 1990.

[2] R. Hojati, S. Krishnan, R. K. Brayton, "Early Quantification and Partitioned Transition Relations," In Proc. of International Conference on Computer-aided Design, 1996.

[3] R. K. Brayton et al., "VIS: A System for Verification and Synthesis," In proceeding of Computer-Aided Verification, 1996.

[4] H. J. Touati, H. Savoj, B. Lin, R. K. Brayton and A. Sangiovanni-Vincentelli, "Implicit State Enumeration of Finite State Machines using BDDs," In Proc. of International Conference on Computer-Aided Design, 1990.

[5] Y. Hong, P. Beerel, J. Burch, K. McMillan, "Safe BDD Minimization Using Don't cares," In Proc. of 34th Design Automation Conference, 1997.

[6] S. Reda, A. Wahba, A. Salem, "M-CHECK: A Multiple Engine Combinational Equivalence Checker," In Proceedings of the International symposium on Circuit and Systems-ISCAS2000, Geneva, 2000.

[7] F. Brglez, D. Bryan, and L. Kozmniski, "Combinational Profiles of Sequential Circuits," In Proceedings of the International Conference on Circuit and Systems, 1989.

[8] T. Shiple, R. Hojati, A. Sangiovanni-Vincentelli, and R. K. Brayton, "Heuristic Minimization of BDDs using don't care," In Proc. of Design Automation Conference, 1994.


```
Forward_Traverse(M)
begin
Qr = Q°; Qn = Qr;
while (true)
 begin
  1. Qn = ∃ q∈ Qn (R(q, q'));
  2. R(q, q')) = Cofactor(R(q, q')) ∨ Qn,
       R(q, q')) ∧ ¬ Qn);
  3. Qn = Qn | q'= q
  4. Qn = Qn ∧ ¬Qr ;
  5. If Qn == ⊥ then break;
  6. Qr = Qr ∨ Qn
 end
```

Figure 9: Final implementation